

Zásobníkové automaty a LL gramatiky

Zásobníkový automat je schopen akceptovat (rozpoznávat) věty tvořené bezkontextovými gramatikami. Je definován uspořádanou sedmicí $(K, T, G, d, q_0, z_0, F)$, kde

- K je množina vnitřních stavů,
- T vstupní abeceda,
- G abeceda zásobníku,
- d zobrazení $K \times (T + \{e\}) \times G^* \rightarrow S$, S je množina konečných podmnožin $(K \times G^*)$,
- q_0 počáteční vnitřní stav automatu.
- z_0 počáteční obsah zásobníku a
- F je množina koncových stavů.

Konfigurace automatu se dá popsat uspořádanou trojicí (q, w, alfa) , kde q je vnitřní stav, w dosud nezpracovaná část vstupu a alfa obsah zásobníku. Na počátku práce je automat v konfiguraci (q_0, w, z_0) .

Zásobníkový automat může akceptovat vstup buď koncovým stavem, nebo stavem zásobníku.

Existují dva základní způsoby analýzy věty - shora dolů a zdola nahoru. Pro analýzu shora dolů potřebujeme zásobníkový automat, kde

- $K = \{q\}$ (automat má jen jeden vnitřní stav),
- T je shodná s množinou terminálních symbolů rozpoznávané gramatiky,
- $G = N+T$, tj. v zásobníku se může vyskytnout jakýkoliv symbol rozpoznávané gramatiky,
- d je dáno rozkladovou tabulkou,
- $q_0 = q$, počáteční stav automatu je q , neboť automat jiné stavy nemá,
- $z_0 = S$, tj. na počátku je v zásobníku startovací symbol gramatiky
- $F = \{\}$, což se interpretuje jako "automat akceptuje vyprázdněním zásobníku".

Analýza zdola nahoru je obecnější a vyžaduje trochu složitější automat:

- $K = \{q, r\}$, stav q je "pracovní", stav r "akceptační",
- T je shodná s množinou terminálních symbolů rozpoznávané gramatiky,
- G je v nejjednodušším případě rovno $N+T+\{\#\}$, tj. sjednocení symbolů gramatiky a speciálního symbolu "#"; deterministický automat může mít množinu G složitější, viz LR gramatiky (cca 12. cvičení),
- d je dáno rozkladovou tabulkou,
- $q_0 = q$,
- $z_0 = \#$,
- $F = \{r\}$.

Analýza řetězce metodou shora dolů

Pro příklad si ukážeme akceptaci řetězce "abaaab" automatem řízeným gramatikou

```
S --> aAS (1)
S --> b (2)
A --> bA (3)
A --> a (4)
```

```
(q, abaaab, S) :- start
(q, abaaab, aAS) :- rozklad podle (1), dále jen (1)
(q, baaab, AS) :- srovnání terminálních symbolů, dále jen srovnání
(q, baaab, bAS) :- (3), rozkládá se vždy nejlevější symbol
(q, aaab, AS) :- srovnání
(q, aaab, aS) :- (4)
(q, aab, S) :- srovnání
(q, aab, aAS) :- (1)
(q, ab, AS) :- srovnání
(q, ab, aS) :- (4)
(q, b, S) :- srovnání
(q, b, b) :- (2)
(q, e, e) akceptováno
```

LL gramatiky

LL gramatiky se používají pro analýzu shora dolů. Písmena LL znamenají, že se vstup čte zleva doprava a že se derivace přepisují rovněž zleva doprava (u LR gramatik - viz příklad na analýzu zdola nahoru - se pravidla redukuje zprava doleva). Jednoduchá LL gramatika je taková gramatika, kde každá pravá strana začíná terminálním symbolem. Navíc musí platit, že pro pravidla $A \rightarrow \dots$ jsou počáteční terminální symboly různé. Tzv. q-gramatika vypadá podobně jako jednoduchá LL gramatika, ale může navíc obsahovat i e-pravidla. Obecná LL(1) gramatika nemá omezení, ale musí pro ni existovat rozkladová tabulka.

Příklady

1. Sestavte rozkladovou tabulku a pokuste se akceptovat řetězce "abbab" a "aaa". Gramatika G je dána:

```
S --> aAS (1)
S --> b (2)
A --> a (3)
A --> bSA (4)
```

Řešení:

M	a	b	e
S	1	2	
A	3	4	
a	sr.		
b		sr.	
e			akc.

V horní řadě tabulky je počáteční symbol vstupujícího řetězce (e znamená, že na vstupu nic není). V levém sloupci je symbol na vrcholu zásobníku. Číslo v tabulce odpovídá přepisovacímu pravidlu, sr. znamená srovnání, akc. akceptaci.

V dalších příkladech nebudeme do konfigurace automatu zapisovat vnitřní stav (je jen jeden). Navíc ale budeme zapisovat seznam přepisovacích pravidel, které jsme při rozpoznávání použili.

```
(abbab, S, e) :- start
(abbab, aAS, 1) :- na vstupu bylo a, na vrcholu S => použijeme pravidlo 1, dále jen (1)
(bbab, AS, 1) :- srovnání
(bbab, bSAS, 14) :- (4)
(bab, SAS, 14) :- srovnání
(bab, bAS, 142) :- (2)
(ab, AS, 142) :- srovnání
(ab, aS, 1423) :- (3)
(b, S, 1423) :- srovnání
(b, b, 14232) :- (2)
(e, e, 14232) :- srovnání, akceptace
```

Nyní zkusíme řetězec "aaa":

```
(aaa, S, e) :- (aaa, aAS, 1) :- (aa, AS, 1) :- (aa, aS, 13) :- (a, S, 13) :-
(a, aAS, 131) :- (e, AS, 131) :- neakceptováno
```

2. Pro gramatiku G sestavte rozkladovou tabulku a akceptujte řetězec "ddbcccc". G =

```
S --> dSA (1)
S --> bAc (2)
A --> dA (3)
A --> c (4)
```

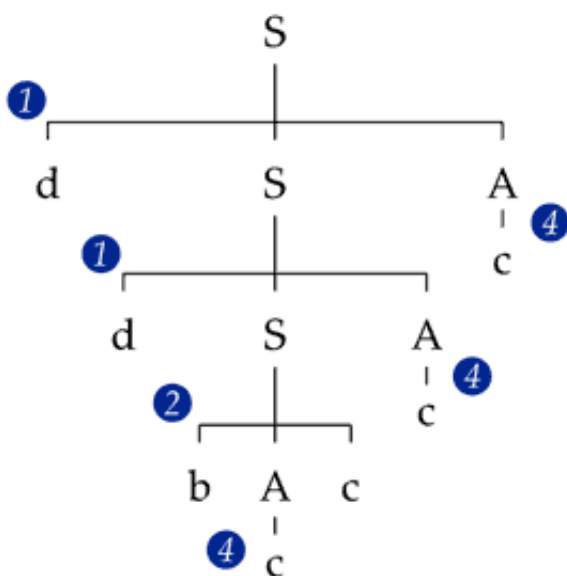
Řešení:

M	b	c	d
S	2		1
A		4	3

V tabulce už jsme nezapsali do levého sloupce terminální symboly, v horní řadě jsme vynechali "e" - tato část je pokaždé stejná a jen zabírá místo. Zápis akceptace také zjednodušíme: bude-li to možné, budeme dva kroky "přepsání neterminálního symbolu" a "srovnání" zapisovat jako jednu činnost.

(ddbcccc, S, e) :- (dbcccc, SA, 1) :- (bcccc, SAA, 11) :- (cccc, AcAA, 112) :-
(ccc, cAA, 1124) :- (cc, AA, 1124) :- (c, A, 11244) :- (e, e, 112444) :- akceptace

Procházíme-li seznam použitých pravidel zleva doprava, můžeme snadno nakreslit derivační strom:



3. Určete rozkladovou tabulku pro q-gramatiku a akceptujte řetězec "aacbb". G je dána:

S --> aAS (1)
S --> b (2)
A --> cAS (3)
A --> e (4)

Řešení:

Vyskytne-li se v gramatice e-pravidlo, v našem případě pravidlo 4, nastává problém. Není totiž zřejmé, zda a kdy použít toto pravidlo k přepisu neterminálního symbolu na vrcholu zásobníku. Je-li na vrcholu symbol "A" a na vstupu symbol "c", použijeme zřejmě přepsání podle (3). Můžeme použít (4) v případě, že je na vstupu "a" nebo "b"? Odpověď je kladná, může-li za symbolem A symbol "a" nebo "b" následovat. K tomu se hodí funkce follow, která je definovaná takto:

follow(Q) = {c: S -->* alfa Q beta, beta -> c gama,
kde c je terminální symbol, S startovací symbol
a alfa, beta, gama řetězce z terminálních i neterminálních symbolů} +
{e: S -->* alfa Q}

Lidově řečeno jsou to ty terminální symboly, které se mohou za daným neterminálním symbolem vyskytnout. Dá-li se startovací symbol přepsat na větu, ve které je daný neterminální symbol až na konci, patří do množiny follow i prázdný řetězec.

Na výpočet funkce follow existují tři postupy:

- o kouknu a vidím,
- o rekurzivní sestup, který je formalizací "kouknu a vidím" a
- o "algoritmus s tečkou", který se dobře implementuje a který si ukážeme později.

Nás bude zajímat $\text{follow}(A)$. První metodou snadno zjistíme, že $\text{follow}(A) = \{a, b\}$. Rozkladová tabulka je tedy:

M	a	b	c
S	1	2	
A	4	4	3

Zkusíme něco akceptovat:

(aacbb, S, e) :- (acbb, AS, 1) :- (acbb, S, 14) :- (cbb, AS, 141) :- (bb, ASS, 1413) :- (bb, SS, 14134) :- (b, S, 141342) :- (e, e, 1413422) :- akceptace

4. Určete rozkladovou tabulku pro q-gramatiku a akceptujte řetězce "aca" a "a". G je dána:

S --> aA (1)
 S --> b (2)
 A --> cSa (3)
 A --> e (4)

Řešení:

$\text{follow}(A) = \{e, a\}$, rozkladová tabulka je proto:

M	a	b	c	e
S	1	2		
A	4		3	4

První řetězec:

(aca, S, e) :- (caa, A, 1) :- (aa, Sa, 13) :- (a, Aa, 131) :- (a, a, 1314) :- (e, e, 1314) :- akceptace

Na druhém řetězci si ukážeme, proč jsme museli do rozkladové tabulky dolpnit sloupec pro "e", resp. proč $\text{follow}(A)$ obsahuje "e":

(a, S, e) :- (a, aA, 1) :- (e, A, 1) :- (e, e, 14) :- akceptace

Kdybychom neměli v rozkladové tabulce zmíněný sloupec, nikdy bychom se z konfigurace (e, A, 1) nedostali!

5. Vytvořte rozkladovou tabulku a akceptujte větu "a+a". $G(N, T, P, E)$ je dána:

E --> TE' (1)
 E' --> +TE' (2)
 E' --> e (3)
 T --> FT' (4)
 T' --> *FT' (5)
 T' --> e (6)
 F --> (E) (7)
 F --> a (8)

Řešení:

G je obecná LL gramatika - není jednoduchá ani q-gramatika, ale rozkladová tabulka existuje. Pro její vytvoření potřebujeme znát první terminální symbol každé pravé strany. K tomu slouží funkce $\text{first}(\alpha)$:

$\text{first}(TE') = \{a, ()\}$

```

first(FT') = {a, (}
first(+TE') = {+}
first(*FT') = {*}
first{(E)} = {(}
first(a) = {a}

```

Výpočet funkce first se dá opět dělat metodou "kouknu a vidím", rekurzivním algoritmem nebo "algoritmem s tečkou". Prozatím se spokojíme s první metodou.

V gramatice jsou dvě e-pravidla, budeme tedy potřebovat i funkci follow:

```

follow(E') = {e, )}
follow(T') = {e, ), +}

```

Sestavit rozkladovou tabulku je nyní prosté:

M	a	+	*	()	e
E	1			1		
E'		2			3	3
T	4			4		
T'		6	5		6	6
F	8			7		

Zkusíme akceptovat "a+a":

```

(a+a, E, e) :- (a+a, TE', 1) :- (a+a, FT'E', 14) :- (a+a, aT'E', 148) :-
(a+a, T'E', 148) :- (a+a, E', 1486) :- (a, TE', 14862) :- (a, FT'E', 148624) :-
(e, T'E', 1486248) :- (e, E', 14862486) :- (e, e, 148624863) :- akceptace

```

Zkusíme si vypočítat funkci first pomocí nerekurzivního algoritmu:

Algoritmus výpočtu first(alfa)

Zjišťujeme, jakým terminálním symbolem začíná řetězec alfa=X1X2...Xn

Krok 1a:

Položíme množinu $F = \{.X_1X_2...X_n\}$

Krok 1b:

Je-li v F pravidlo $B \rightarrow \beta$, přidáme do F pravidla $A \rightarrow \beta$.
Tento krok provádíme tak dlouho, dokud to jde.

Krok 1c:

Je-li v F pravidlo $B \rightarrow \beta$, dáme do F pravidla z F, ve kterých se vyskytovaly symboly β , ale tečku umístíme až za B

Krok 1d:

Kroky b a c se opakují tak dlouho, dokud lze do F přidávat.

Krok 2:

$first(alfa) = \{ \text{všechny terminální symboly z F, které jsou bezprostředně za tečkou} \}$
 $+ \{e: \text{je-li tečka na konci pravidla} \}$

Zkusíme si spočítat např. $\text{first}(E)$:

$F = \{ .E$	krok 1a; množina F a neterminální symbol F nemají nic společného, je to jen shoda jmen
$E \rightarrow .TE'$	
$T \rightarrow .FT'$	
$F \rightarrow .(E)$	
$F \rightarrow .a$	předchozí čtyři pravidla se přidala na základě kroku 1b
$\}$	
$\text{first}(E) = \{a, (\}$	krok 2

6. Vypočtěte $\text{follow}(A)$ v gramatice G :

$S \rightarrow aSAb$
 $S \rightarrow AB$
 $A \rightarrow Bb$
 $A \rightarrow aA$
 $B \rightarrow bB$
 $B \rightarrow e$

Řešení:

Algoritmus výpočtu $\text{follow}(A)$

Krok 1:

Položíme N_e rovno množině všech prvků, které se dají přepsat (i rekurzivně) na prázdný řetězec.

Krok 2a:

$F = \{ A \rightarrow A. \}$, do F umístíme fiktivní pravidlo, z kterého vyjdeme.

Krok 2b:

Je-li v F pravidlo $B \rightarrow \text{gamma}.$, kde gamma je neprázdný řetězec, dáme do F všechna pravidla, ve kterých je na pravé straně B a tečku umístíme za B . V podstatě budeme dále určovat $\text{first}(\text{řetězec, který následuje za } B)$.

Krok 2c:

Je-li v F pravidlo $C \rightarrow \text{alfa}. B \text{ beta}$, přidáme do F všechna pravidla s B na levé straně, tečku umístíme na začátek.

Krok 2d:

Je-li v F pravidlo, kde je za tečkou neterminální symbol, který patří do N_e , do F vložíme toto pravidlo ještě jednou, ale tečku posuneme o jeden symbol doprava.

Krok 2e:

Kroky 2b, 2c, 2d opakujeme, dokud do F můžeme přidávat další položky.

Krok 3:

Do $\text{follow}(A)$ dáme všechny terminální symboly, před kterými je tečka. Je-li v F pravidlo $S \rightarrow \text{alfa}.$, kde S je startovací symbol, přidáme do $\text{follow}(A)$ i symbol "e".

Aplikujeme algoritmus na výpočet $\text{follow}(A)$:

$N_e = \{B\}$

$F = \{ A \rightarrow A. \quad \text{krok 2a}$
 $S \rightarrow aSA.b$
 $S \rightarrow A.B \quad \text{krok 2b (A je na pravé straně)}$
 $A \rightarrow aA.$

 $B \rightarrow .bB \quad \text{krok 2c}$
 $B \rightarrow . \quad \text{(plyne z } S \rightarrow A.B)$

 $S \rightarrow AB. \quad \text{krok 2d (plyne z } S \rightarrow A.B)$

 $S \rightarrow aS.Ab \quad \text{krok 2b (plyne z } S \rightarrow AB.)$

 $A \rightarrow .Bb \quad \text{krok 2c}$
 $A \rightarrow .aA \quad \text{(plyne z } S \rightarrow aS.Ab)$

 $A \rightarrow B.b \quad \text{krok 2d (plyne z } A \rightarrow .Bb)$
 $\}$

$\text{follow}(A) = \{e, a, b\}$ (e proto, že je v F pravidlo $S \rightarrow AB.$)

7. Sestavte rozkladovou tabulku a akceptujte řetězec "[[[]] []]" pro G:

$S \rightarrow [S]S \quad (1)$
 $S \rightarrow e \quad (2)$

Řešení:

Je zřejmé, že $\text{follow}(S) = \{e, \}$. Proto bude rozkladová tabulka:

M	[]	e
S	1	2	2

Akceptujeme řetězec:

$([[[]][[]], S, e) :- ([[]][[]], S]S, 1) :- ([[]][[]], S]S]S, 11) :- ([[]][[]], S]S]S]S, 111) :-$
 $([]][[]],]S]S]S, 1112) :- ([[]], S]S]S, 1112) :- ([[]],]S]S, 11122) :- ([[]], S]S, 111222) :-$
 $([]], S]S]S, 111221) :- ([],]S]S, 1112212) :- ([], S]S, 1112212) :- ([],]S, 11122122) :-$
 $(e, S, 11122122) :- (e, e, 111221222) :- \text{akceptace}$

8. Vytvořte LL(1) gramatiku, která akceptuje výraz $1^n a 0^n$, kde n je celé číslo a vyjadřuje počet opakování symbolů 0 a 1. Vytvořte rozkladovou tabulku a akceptujte "11a00".

Řešení:

$S \rightarrow a \quad (1)$
 $S \rightarrow 1S0 \quad (2)$

Rozkladová tabulka:

M	0	1	a
S		2	1

Akceptujeme řetězec:

$(11a00, S, e) :- (1a00, S0, 2) :- (a00, S00, 22) :- (00, 00, 221) :- \text{akceptace}$